A Distributed Collaborative System for Flexible Learning Content Production and Management

Abelardo Pardo, Jesus Arias Fisteus and Carlos Delgado Kloos

Department of Telematic Engineering, University Carlos III of Madrid
Avenida Universidad 30, Leganés E-28911 (Madrid) Spain
Email: abel@it.uc3m.es

Authoring learning content is an area under pressure due to conflicting requirements. Adaptive, template-based, highly interactive, multimedia-rich content is desired for current learning environments. At the same time, authors need a system supporting collaboration, easy re-purposing, and continuous updates with a lower adoption barrier to keep the production process simple, specially for high enrollment learning scenarios. Other areas such as software development have adopted effective methodologies to cope with a similar increase in complexity. In this paper an authoring system is presented to support a community of authors in the creation of learning content. A set of pre-defined production rules and templates are offered. Following the single source approach, authors create documents that are then automatically processed to obtain various derived resources. The toolkit allows for simple continuous updates, the re-use and re-purpose of course material, as well as the adaptation of resources to different target groups and scenarios. The toolkit has been validated by analyzing its use over a three year period in two high enrollment engineering courses. The results show effective support and simplification of the production process as well as its sustainability over time.

ACM Classification: *K.3.1, J.1*

1. Introduction

Learning content is at the heart of all learning activities. At a 2005 survey about content authoring research conducted by The eLearning Guild, 78 of the participants agreed that "Content is King" was considered as conventional wisdom (Pulichino, 2005). And yet, in the same study, 68 of the participants also agreed that content collection and production was the least understood, least-often addressed and least studied of all e-learning elements. Learning content has been under the influence of the latest trends in Internet content, namely, they are becoming highly interactive, multi-format, multimedia-based, created collaboratively, etc. But on top of these trends, learning material has three more aspects that make it more complex to manage than regular Internet content: the material needs to be adapted to multiple learning scenarios, students, target groups, etc; there is a high level of re-use of low granularity portions of the material; the content is subject to continuous updates. The combination of all these factors when producing and managing e-learning content translates into a hefty increase in the process complexity. Authoring environments need to bridge the gap between an increasingly sophisticated final product and a community of users that demand a low adoption threshold. In the previously mentioned study,

Manuscript received: 31 December 2011
Communicating Editor: José-Luis Sierra-Rodríguez

the three features that users wanted to be enhanced in the authoring tools were: content re-use and re-purposing, more instructional design process, and rapid development templates. Two of these three aspects (re-purposing, and templates for rapid development) have been successfully addressed in the area of software development.

In general, designing high-quality learning resources usually comes at the expense of a costly production flow. The situation is more critical if this activity is part of a more general learning experience (for example, a course) where a high number of activities are created by a team of authors working collaboratively and concurrently. Creating and maintaining such a potentially large set of resources over a large period of time requires an extremely streamlined production flow to reduce its cost. Conventional office-suite tools do not withstand the requirements of such environments. Although they can be used to produce some resources, they are not conceived to support authoring of more sophisticated resources in a collaborative environment. Several technological approaches have been proposed to tackle this problem. Single source techniques create multiple views of a single document. Additionally, markup languages allow documents to include content and information about their structure. Tools exist that are invoked programmatically and therefore become a piece in a more complex work flow. Other tools such as version control systems have also evolved to accommodate powerful and robust collaborative work patterns.

But the effective combination of all these technological advantages and their deployment in a sustainable real authoring environment still remains unsolved. In this document the "Adagio" system is presented. It allows the definition of production rules to combine already existing tools to offer a learning content production and management platform that simplifies the adaptation of material to various target groups, allows the re-use and re-purpose of material at any level of granularity, and supports a continuous update process, all in a collaborative environment. The system has been in use during the last three years by two author communities producing content for two high enrollment blended-learning engineering courses. By analyzing the production flows in these communities we conclude that the proposed system offers a realistic, sustainable, and flexible authoring support[1].

The rest of the document is organized as follows. Section 2 includes a description of the main contributions in the area of distributed learning content creation. The scenario considered for the proposed system is described in Section 3. Section 4 describes the production rules offered by the system as well as their use in the described scenario. Section 5 describes how the system provides support for various course aspects that emerged during its enactment. The description of the proposed distributed production work flow is included in Section 6. The validation of the proposed system, shown in Section 7, is performed by a detailed analysis of two authoring communities where the system is currently in use. Finally, a set of conclusions and lines for future work are described in Section 8.

2. State of the Art

Authoring platforms have been present from the early steps of computer systems and have played an important role in the context of instructional design. In 1986, O'Neil described how authoring systems and languages were evolving from the 70s toward providing more sophisticated support to process highly structured and "componentised" content structures, and provide support for

1 The system is available at http://www.it.uc3m.es/abel/Adagio/FAQ.html

instructional decisions (O'Neal, 2008). The need for authoring systems that capture the structure of course material as well as its role within the instructional design has prompted the appearance of several approaches as shown in Rawlings *et al* (2002). But these systems are solely concerned with how resources are sequenced during a learning experience, leaving still open the previous problem of how to produce large collections of resources that later need to be properly orchestrated.

Descriptive markup systems were identified as a powerful paradigm that would accelerate the production pace of resources by focusing only on content and not on appearance (Coombs *et al*, 1987). The idea is to create both the content and a set of marks that identify its parts. The so-called "single source" schemes are based on the premise that multiple views can be generated from a single source document using markup languages and a set of tools for post-processing. This approach has been widely adopted in the production of technical documentation and marginally used for learning material (Molloy, 2003; Thomas and Ras, 2005; Thomas and Trapp, 2007; Walsh, 2007). But one of the main problems of these systems is to provide simple re-use and extensibility for re-purposing and adapting material to new formats and environments, specially in large scale scenarios.

Sierra *et al* (2005, 2006) describe in detail the contexts in which there is an intensive use of information provided by experts. Education is identified as one of these domains in which authors handle highly structured documents about a very specific topic. Descriptive markup technologies were identified as the adequate support to use because the representation can be considered human-readable and can also be efficiently processed. The adoption of this technology typically translates into an initial increase in production complexity that turns into a lower maintenance and higher portability in the long run.

But when using markup technologies, a trade off appears between semantic and purely structural markups. Fisler *et al* (2005) and Fisler and Schneider (2009) present two open-source e-learning tools, a content management system and a tool that uses the specifically designed markup language eLML to capture the structure of course material. This is just one sample of the existing approaches that aim at "normalizing" the structure of a course. The authors justify the use of an ad-hoc language on the need for some strict didactic and technical structures to create similar looking lessons. In the same work, the authors also acknowledge the importance of avoiding rigidity in the adopted markup and the definition of rules for automatic production.

Another example of the use of markup languages to create learning material is within the context of the "Connexions" platform[2]. Connexions is an open space where authors can create and share educational material organized in units called "modules". This initiative proposes an approach to content publishing which is also based on markup languages (Henry, 2004). The need for fast paced communication is the main reason for education to incorporate new materials easily into the courses. The project proposes the use of the XML-based language CNXML to mark the course material using visual editors. A production process is then applied to obtain the final resource for publication. The proposed authoring paradigm is also described as ideal for long-term viability (Dholakia *et al*, 2006). The modularity of course development is also at the basis of other approaches using markup languages (Baudry *et al*, 2004). The ultimate goal is to achieve a construction-kit style in which learning material is built from a set of previously created "bricks" (Bungenstock *et al*, 2002).

2 http://cnx.org

Docbook (Stayton, 2007) is another powerful markup language for content structured in chapters, sections, paragraphs, etc. Its use for e-learning content production has been considered in other initiatives as described by González Téllez (2010), where material is described in Docbook and then packaged following the IMS Content Packaging specification. As in the previous scenarios, using a markup language clearly poses an advantage in production environments in which content needs to be manipulated to comply with various restrictions. The work described in this document assumes the use of Docbook to create course material, which is then automatically processed by a set of rules to produce different type of resources. With these rules, the ever increasing type of resources can be easily accommodated in the production flow.

The change on the type of material used in learning environments is obviously affecting the production stage. From a text-book based learning in which resources are perceived by students as "read-only" and by instructors as "write-once", we are moving to scenarios in which students have a more active role and, as a consequence, content needs to be quickly updated, but at the same time must maintain a coherent style (Hadzilacos *et al*, 2007). In this context, it is very important to offer authoring platforms where material can be easily managed by a set of authors and updated with almost no effort. As pointed out by Errington (2004), the possibilities offered to teachers have an impact on their beliefs, which are then essential to adopt new paradigms in their courses.

Collaboration is another factor of content creation that is gaining importance. Learning content is created increasingly by teams of authors working collaboratively over a large set of documents. The success of collaborative creation is well documented in other areas such as Free/Libre Open Source Software (FLOSS). Unfortunately, although FLOSS communities are a good example of successful open participatory learning ecosystems, resource creation is one of the aspects that still needs significant improvement as pointed out by Meiszner *et al* (2008). There have been some initiatives in the past to deploy the FLOSS model in the context of learning, but they have had limited impact (González-barahona *et al*, 2005; Barahona *et al*, 2005).

The work presented in this document is similar in spirit to the one presented by Martínez-Ortiz *et al* (2006). The adopted scheme is based on the "manual writing metaphor" because it is less knowledge demanding for authors. The work flow proposed in this work enhances the production process by combining a set of versatile rules with basic pedagogical patterns to support re-use and re-purposing, allow easy adaptation to new environments, and integrate these features in a fully collaborative environment.

Product process automation is a discipline well studied and widely used in the area of software engineering. Tools for automatic software build such as Make, Ant or Maven (to mention just a few), and techniques such as "continuous integration" (Fowler, 2006 accessed 12 April 2012) are commonly used in software development environments. Although the requirements derived from authoring educational material are similar, there are two important issues that make these existing tools unsuitable for managing the process. The first one is the highly specialized and rigid context in which they have to be deployed. Aspects such as automated testing, alpha releases, self-tests, etc. are not needed in an authoring environment. Also, a collection of educational resources is different from a software product that needs to follow a successful building process that is then compared to a set of tests. And second, these applications typically require target users to have significant knowledge about the intricacies of the process. Authoring of educational resources should impose a minimal footprint in the authoring space so that users with no technical background can still use it. Thus, the trade off that needs to be explored is how to offer the maximum functions for flexible content production while keeping the footprint at its minimum.

The proposed system, "Adagio", assumes that several authors are already using various tools to create the basic building blocks of a large set of learning resources. In this scenario, a distributed environment is provided for the collaborative production and combination of a repository of learning content. Through the use of the version control system "Git", authors work in their personal spaces with the source files of a collection of resources that are stored as a distributed file system. Each folder containing a resource is extended with a set of production rules to automatically process and combine resources and create a global repository. A basic set of rules to process the resources is defined including a powerful system to extend these rules. A concrete work flow is described where the role "master publisher" is proposed as the person in charge of collecting the changes produced by the authors, creating a new version of the resources and publishing them in the final location. The objectives of the system are to achieve the lowest possible latency when propagating changes, facilitate re-purposing of material, allow the adaptation of course material to various formats and target groups, support a flexible community of authors, and achieve sustainability over long periods of time.

3. Scenario for Distributed Production of Learning Content

The Adagio platform has been designed for scenarios of high-enrollment courses (with hundreds of students), a large community of instructors (even more than 10) and hundreds of possibly multilingual published resources. Nevertheless, it also fits simpler and smaller learning scenarios.

In this document, by production we understand the steps from the conception of the course structure until the final set of documents is ready to be published in the virtual communities where students access them. The first aspect to be considered is the pedagogical strategy adopted. The system supporting this process should strive to be an "authoring system" and not simply an "authoring language" (O'Neal, 2008).

Typically, a wide variety of resources need to be produced in a course. For example, some courses in which Adagio is being used publish administrative documents, course syllabus, class notes, lecture slides, weekly activities for the laboratory, homework, etc. These documents are collaboratively produced by various instructors in several formats (PDF and PowerPoint for slides, HTML for activities, etc.) from different types of source documents (Microsoft Word and PowerPoint, Docbook, Latex, etc.) The authoring platform, despite having a set of preferred formats, has to support legacy formats in order to simplify its adoption by instructors. Courses taught in more than one language pose an additional challenge to the course material production and publishing stages.

Being able to produce different views from the same source document is an approach that significantly reduces the complexity of the production process. These views are used to adapt the materials to: different visualization platforms, such as mobile devices, RSS channels and paper, which requires a printer-friendly version, typically in PDF format; different audiences, such as special versions of activities for instructors, which include solutions, lecture notes, etc. that students cannot see; different instants of time, such as activities that include solutions for students to review, but only after a given amount of time passed since the activity was first published.

Due to the large number of instructors that need to collaborate in the production of materials, the platform needs to support effective collaboration work flows, including concurrent editing of materials, version control, quality control, etc.

Content updates need also to be managed throughout the course. On the one hand, some courses

are based on static materials that are available from the beginning to the end of the course, and do not need special handling. On the other hand, other courses are based on "just-in-time" teaching schemes (Novak *et al*, 1999) where students work in the course material before attending the class, the instructor receives feedback about this process and adjusts the upcoming session material accordingly. This translates into a set of resources that need to be modified or updated during the course enactment. For example, let us consider an activity where students are asked to fill-out their profile in the virtual community and instructors then create a summary document (Bullard and Felder, 2007). This type of activities require agile content management to insert new documents as part of the collection of course resources. In other words, if active learning is adopted in a course, it is likely that the course material needs to be frequently modified.

But even in those cases in which course material can be considered "highly stable", there is always the need to make corrections, amendments, etc. For the case of a large community of authors, with a large variety of documents and the requirement to have bilingual resources, the need for a streamlined resource management procedure arises.

Once the course material has been produced and deployed, the adopted work flow needs to be sustainable across course editions. This sustainability basically means two things. Firstly, the course material should be easy to re-use in future editions of the course, even if the course is slightly changed. This includes the ability to maintain repositories of reusable materials from which individual contents can be picked to compose lecture guides, problem sets, etc. for specific course editions. Secondly, the community of authors could also change, and new instructors should be brought into the production cycle as quickly as possible.

Last but not least, as different courses may have quite different requirements, the platform should be extensible so that tailor-made components can be developed and deployed by course administrators when their requirements are not fully fulfilled by the platform.

The proposed authoring platform has been conceived to tackle the issues arising in these scenarios. The system specifications were derived from previous experiences on managing large enrollment courses with a sizable set of instructors and resources. The triggering factor that prompted the design and deployment was the use of bilingual course notes. Every resource in the course needed to be translated and both versions should be maintained in parallel. The requirements were derived mainly to simplify the role of the person in charge of merging the different contributions of the authors with occasional input of the authors. Conventional authoring tools focus on supporting authors in the creation of single resources, but fail to support the production and management of a large set of resources for an also large community of authors. Adagio offers a formalism where complex production flows can be captured by sets of rules that automatically derive the resources in their various formats from the source documents.

4. Production Rules

Adagio assumes that the source files for content production are organized into folders, and the production process is decomposed into simpler operations applied to each folder. The operations in a folder may depend on the execution of operations in other folders. The system is based on a set of production rules previously described and the presence in a folder requiring processing of a plain text file, called the "rule" file (by default with name Properties.dgo), where the sequence of production rules is declared. A rule is a pre-defined procedure receiving a set of parameters and applied to a set of entities. Figure 1 shows how a generic rule is invoked in the rule file. The

first line contains within square brackets the rule category (that unequivocally identifies the rule), optionally followed by a dot separated set of hierarchical rule names. This line is followed by a (potentially empty) set of assignments to the rule parameters.

```
[rulecategory.name]
parameter1 = valueA valueB valueC
parameter2 = value1 value2 value3
...
```

Figure 1: A generic production rule

The rule file format is based on the "INI" format used in the Microsoft Windows^{tm} operating system. INI files are human-readable and provide a simple yet powerful description mechanism. A DEFAULT rule can be defined containing only variable definitions that can be used in other rules. Any rule can refer to any previously defined variable, which is replaced by its value when the file is processed. Figure 2 shows a rule file with a variable declaration in the default rule and its use in a rule to process XML documents.

```
[DEFAULT]
styles = style/dir
[xslt]
files = source.xml
styles = %(styles)s/translate.xsl
```

Figure 2: Rule file with variable declaration

Every parameter of every rule in Adagio has a default value. Thus, a high customization level is possible while maintaining simple invocations for typical operations. Aside from its specific variables, every rule in Adagio has the following parameters always defined:

- Source and destination folders. The location from where the source files are taken, and the newly produced resources are created.
- Time of visibility. Two date/time instances defining the time interval when the rule is executed. This feature is useful to produce resources only during certain time frames.
- Rule enable. The rule is disabled when set to "false" or "off".
- Rule permission. A string so that when present in a set of permissions (or empty), the rule is executed.
- Help. Text of arbitrary length describing the functionality implemented by the rule. Adagio may be invoked to display its content, thus behaving as a self-documented application.
- Languages. A space separated list of languages, used by those rules capable of producing multiple versions of a document depending on the language.

An author creates in a local folder the basic documents to produce the resources that are part of a larger project, creates the rule file with the appropriate parameters, and invokes its execution by double clicking in the file icon, or executing the "adagio" command in an interpreter.

4.1 Predefined Production Rules

Adagio includes 17 production rules, divided into six categories, to support the most common operations identified in a production flow. Each rule contains a set of parameters to control execution aspects. The six categories are:

- XSLT processing. Rules to apply XSL transformations to XML documents. A comprehensive set of transformations are included to process files written in Docbook (Stayton, 2007). These transformations assume a single source paradigm and derive multiple documents by selecting the appropriately marked portions. These transformations are sensitive to special markups to support the following functionalities:
  - Multi-language. Various translations of the text co-exist in the document and are segregated by the production rules when specified in the "languages" variable.
  - Multiple target users. Documents contain regular content (no special markup), solutions, and instructor guide content. The rules generate three separated resources selecting each document subset.
  - Exams and quizzes. Transformations to produce open question exams and quizzes. Generation of multiple exam versions through question shuffling, grading templates, and solution sheets are also supported.
  - Submission form. When the submission of a document is required, the form is automatically derived.
  - Multimedia embedding. Video, screen captures, etc.
- Image manipulation. Rules for basic image transformations and format exchange.
- File transfer. Rules to copy and move files to specific locations. Massive file transmissions to remote locations, for example for being published in a Web server, are supported by the "rsync" rule, which uses the Rsync application (Tridgell and Mackerras, 1996).
- Recursive invocation. Adagio is invoked recursively in another folder and its resources optionally exported to the current location. This rule allows, for example, the recursive construction of an entire site by executing the top-level rule file.
- LaTeX and PDF typesetting. Rules to process documents written in LaTeX and produce PDF documents.
- Office documents. Rules to translate office tool suite documents to PDF.
- Extension Rule. A special rule to invoke the execution of an arbitrary script written in the Python programming language (Van Rossum Last accessed: 2011). With this feature, any additional functionality desired in Adagio can be included with a script complying with a simple programming interface.

These rules are designed with a dependency mechanism to automatically detect which rules need to be executed due to modifications in the corresponding source files, thus allowing an efficient support for continuous updates.

4.2 Rule Hierarchy and Re-use

When managing large repositories, production rules tend to have a high degree of similarity. In many folders, most of the production rules are the same or almost the same. Two mechanisms are included in the toolkit to avoid the repetition of large blocks of rules in rule files.

The first mechanism is the possibility of deriving rules from other rules, so that parameter values are inherited. A new rule is derived from a basic one by adding to its name a dot and another name. For example, the "inkscape" rule translates all files in SVG[3] format to PNG. An additional rule to translate the same files to the JPEG format can be derived by creating a rule named "inkscape.to_jpeg" that assigns its "output_format" parameter to "jpeg". All the other parameters are automatically inherited from the "inkscape" rule.

Complex production flows typically require a set of common rules that are re-used in almost all rule files. The best strategy is to define these rules in a common location and make them available to any rule file in the project. The second mechanism supports this functionality by combining various features: a special rule file at the top level of the course, which is always read before the rule file in the current folder; and the "template" keyword, which performs in-place inclusion of other rule files. With these two features, rules can be defined at the course level in a modular way, so that they can be invoked by name from rule files in specific folders, without the need to replicate their definitions. The possibility of defining aliases for rules further simplifies rule files, as rules can be given meaningful names such as "create_lab_session". With this approach, the rule files in each folder are greatly simplified, including just invocations to the rules to be used and, sometimes, some parameter definitions needed to customize those rules.

5. Extensibility

As different courses have different needs, Adagio has been designed with the aim of being flexible to accommodate them. Courses in Adagio are not just a collection of teaching materials. They can contain software that makes them programmable platforms. This software extends Adagio with features tailored for specific courses, and allows things such as widgets that interact with Web servers for sending or receiving information from the material's Web pages, adding support for embedding new types of contents in materials, etc. These are the main mechanisms that allow course administrators to develop custom extensions:

- Custom XSLT style sheets: XSLT is not just a template language. In combination with its function libraries, custom style sheets can be used to process input Docbook or XML documents in powerful ways. For example, by using the role attribute of Docbook, it is possible to do custom formatting of certain fragments of a Docbook document, whilst the rest of the document is processed with the general Docbook templates.
- Custom rules: Adagio provides an API for the development of new processing rules with the Python programming language.

In neither case extensions need to be integrated in the Adagio source tree. Extensions are available to a course just by placing them in the appropriate location of the course repository. Any author having access to the course repository has them available. To illustrate the ability to extend Adagio, the following sections explain some extensions developed for production environments in which Adagio is being used. These extensions were derived from the specific needs appearing in concrete learning scenarios.

5.1 RSS News Feeds

This extension allows publishing a news feed for the course. The feed can be published, by using the same source document, in a RSS channel and in the course Web pages. The extension was

3 Scalable Vector Graphics, www.w3.org/Graphics/SVG

developed by using only XSLT style sheets.

The items to be published in the feed are written in a Docbook file, using the regular Docbook markup and a special role value to delimit them. A style sheet extracts the information from the Docbook file and generates the RSS version of it. The style sheet is able to publish just the most recent items (the number of items can be configured) and automatically set the build time stamp.

The same items, from the same Docbook file, can additionally be published in a course Web page by inserting a special tag in the page and processing it with a style sheet that replaces the tag by the most recent items from the Docbook document the tag points to. Two formats can be selected from the tag: a short version that only shows the first paragraph of the most recent items, and a long version that shows the full content of all the items. The first option is useful for the main page of the course, where a reduced amount of space is available.

5.2 Task Duration Feedback Widget and Statistics

The instructors of a course wanted every published exercise to have a widget that students could use to send feedback about the amount of time they needed to complete the exercise.

The extension was developed with an XSLT style sheet. It matches an element with a given role attribute in the Docbook file of the exercise, and produces the HTML code that renders the widget as a form. The server that receives the information from that form is outside the scope of Adagio. With a custom rule written in Python, data received by the server is analyzed and plots showing times needed by students for each exercise are created and published, with access only to instructors. These plots are generated and created from the course repository itself using Adagio.

This functionality is also an example of how Adagio can be extended by technical personnel not related to the project. This extension was created by an author with technical background who had minimum knowledge about the internal structure of Adagio. The integration of third-party agents is supported in Adagio by offering a generic rule that is able to invoke a user-provided script. There is a set of basic variables that are passed by default to the given script, and the possibility of including a set of arbitrary arguments as additional parameters. With these values, any script can then be easily integrated as a new rule in the system.

5.3 Instructor Comments

In order to facilitate collaboration between the instructors of a course, an extension that allowed them to comment on published materials was developed. An XSLT style sheet parses a given role attribute in a content item (exercise, lecture guide, etc.) and generates a form for it, accessible only from the instructor's version of the page (i.e., not available in the student's version). Instructors can fill in the form with their comments and send them to a server. By interacting with the version control system (see Section 6), the server writes the comments it receives in a file that is included from the Docbook sources of the exercise or lecture guide, so that they are shown in the instructor's version the next time materials are updated.

The extensions described in this section show how the work flow described with Adagio can be easily modified to adopt new requirements in the material. This feature is becoming increasingly important in emerging learning scenarios where resources are no longer being produced nor offered to students in a single location.

6. Distributed Production Work Flow

The power of the described authoring system becomes most effective when used in a large community of authors in combination with a distributed production work flow. This type of work flows have been used for a long time in software development projects (Brooks, 1995), but their application to authoring environments with a large number of resources is not straightforward. Version control systems (or VCS) are the tools typically used for exchanging documents among a set of users working on a project. There are numerous tools currently available, both open source (CVS, Subversion, Git, Bazaar, Mercurial, etc.) and commercial (BitKeeper, Plastic, etc.)

The tool selected to be used in combination with the rules previously described is Git (Chacon, 2009) because it is open-source, and based on a distributed file system in which there is no need for a central server. Although the tool is designed to deal with source code, the percentage of binary files, their ratio of changes, and the requirements on disk space did not pose any complication. Figure 3 illustrates the adopted work flow.

All authors are assumed to have two working areas. A local folder where changes are staged and a "repository" available to the rest of the authors. All authors contribute equally to the production flow except one, called the "master publisher". Upon joining the team, each author clones the resources stored in the master publisher's repository. Once a set of changes has been produced, its author makes these changes available through the local repository and the master publisher is notified. The master publisher is in charge of merging changes proposed by authors, generating and publishing the new versions of the resources, and making those changes available for the rest of authors. This work flow is based on the "Integration-Manager Work flow" model described by Chacon (2009).

The reason to choose this scheme is to offer a self-contained repository to every author. With this scheme, authors may work in resources and create new versions even without an Internet connection, because the repository is stored locally in their computer. The submission of changes to the repository managed by the master publisher can be done at any time. A second advantage of this approach is to provide data integrity through redundancy. Every author space contains a complete copy of the repository (including its history of changes). In the case of a major data loss, resources can be easily recovered using author repositories.

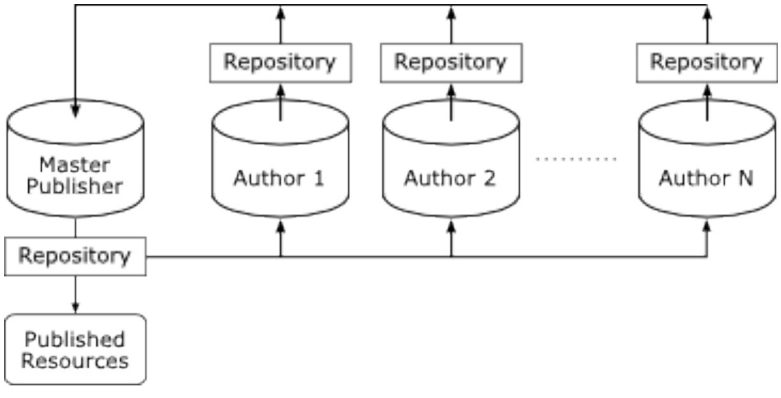

Figure 3: Version control system structure to deploy the work flow

This work flow also opens the possibility to incorporate students into the production flow. The interaction between authors and the master publisher can perfectly accommodate students creating material. Git allows the master publisher to review the changes introduced by students before merging them into the official version of the course materials. In addition, Adagio can distribute the materials of a course across several Git repositories. Thus, the master publisher can grant the students access to only a subset of the content. The requirement for this paradigm would be, though, for students to install the minimum set of tools to create the material (illustration processing, XSLT style sheets for processing Docbook documents, etc.)

The work flow previously described is just one possibility among the multiple options offered by the version control system chosen. The distributed nature and the low level operations offered by Git allow a wide variety of production flows. For example, the role of the master publisher can be removed in favor of a fully distributed management among the authors. After a common structure of the material is described at a higher level, authors may proceed to create the required resources and populate in a totally distributed fashion the course repository. Additional features such as automatic notifications are needed to maintain the awareness of the team of authors about the overall creation process.

7. Experimental Results

Adagio has been in use during the last three years in the production cycle of educational material for two engineering courses offered to over five hundred students per year. In both scenarios, a large number of authors participates in the collaborative production of numerous learning resources. The institution required all course material to be created in English and Spanish. The course content has been managed with the production flow described in Section 6.

The scenarios where Adagio offers an advantage over conventional authoring strategies is when the number of authors and resources to manage is high. Although a formal measure of the threshold over which the tools offer a clear advantage is not feasible, the experience over the last three years offers some insight. The trade off to explore is when the complexity of the overall production process compensates the requirements imposed by Adagio. For teams of more than six authors, the production process already requires a significant exchange of documents and a basic organization. This threshold could be lower if there is a high rotation factor (different authors join and leave the community frequently). A similar approximation can be done with the size of the repository in terms of the number of resources. Repositories with items in the order of several hundreds become difficult to manage, need to be carefully organized, and automatic processing tools like Adagio offer a clear advantage.

The examples described in the remainder of this section comply with these two conditions. They have a large community of authors, and they manage a large number of resources. It follows a description of this process in the two scenarios.

7.1 Case 1: First Year System Programming Course

The first scenario is a first year course "System Programming" of the Telecommunication Engineering program. The numbers of the course material production process are shown in Table 1.

Of the 20 authors participating in the production tasks, only the master publisher contributed to the development of Adagio. The rest of authors were regular users that provided feedback about their experience. During the two and a half year period, there have been three editions of the

Deployment period	Jan. 09 – Jul. 11	HTML Documents	254
Final number of files	3417	Slide Sets	23
Commits in 2009	884	Java Code Files	287
Commits in 2010	471	Pre-compiled code	8
Commits in 2011	325	Figures	73
Total umber of commits	1680	Style resources	65
Number of authors	20	Auxiliary resources	28
Total Resources	765		

Table 1: Activity and resources for the Systems Programming Course

course. The number of authors changed significantly from the 2009 to the 2010 edition. The evolution of the repository is shown in Figure 4.

The left side of the figure shows two clearly defined periods during which the course preparation required a high level of interaction among the authors. The second and third editions (Spring 2010 and 2011) required a lower level of interaction because most of the content from the first edition was re-used. The right side of the figure shows the evolution of the number of files in the repository. As it can be seen, the number of source files is large and keeps growing without showing any limiting behaviour.

In the editions of the course previous to the use of Adagio, materials were edited directly in HTML. For example, in the student guides for the laboratory sessions, a single HTML page contained several independent exercises. A separated repository was kept for the students supervised by each senior lecturer. Although many resources were shared among several of these repositories, they were merely copied and pasted. In subsequent editions, these copied materials began to diverge as they were corrected and improved by different instructors.

Adagio, in combination with Git, greatly simplified the production process. The main improvements were in: the ability to edit self-contained materials that could be automatically

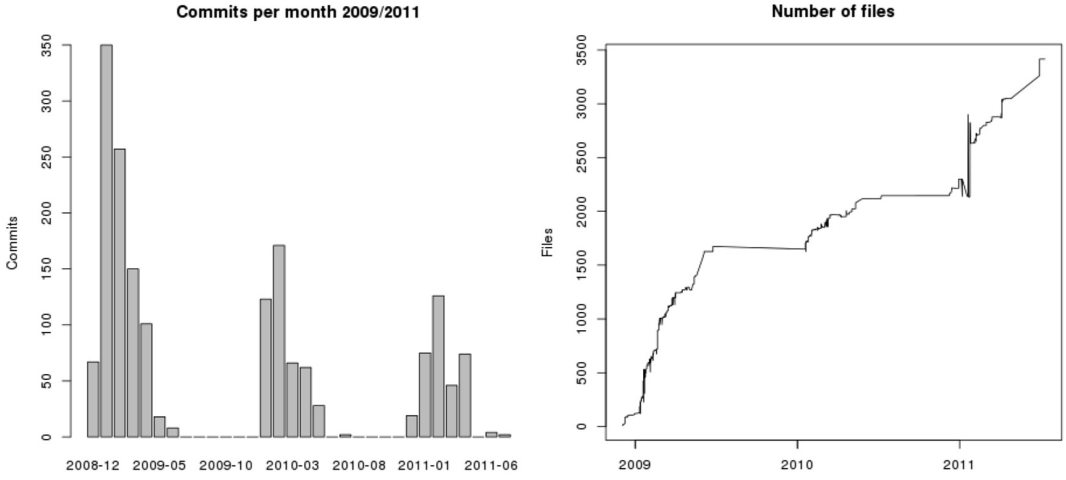

Figure 4: Evolution of the course material repository

included in the class guides published to the students, the unified system for processing different document types (slides, class notes, instructor guide, etc.), and a more effective way of collaboration in the creation of materials (for example, improvements in materials done by one instructor were automatically available to the others).

7.2 Case 2: Second Year Systems Architecture Course

The second scenario to measure the impact of the tool is the third semester course "Systems Architecture", part of the Telecommunication Engineering program[4]. The numbers of the course material production process are shown in Table 2.

The master publisher in this course was the main developer of Adagio. The rest of authors, as in the previous scenario, were regular users offering occasional feedback about their experience with the tool set. The number of authors fluctuated significantly over the three year period. It started with six authors in the first edition and increased up to ten in the third year. The evolution of the repository is shown in Figure 5. The left side of the figure shows the number of changes received in the master repository and, as expected, it is higher during the teaching period and lower between editions. At its peak, the production flow managed a bit over 400 changes per month. These numbers show the power of the adopted distributed approach. The right side of the figure shows the evolution of the number of files in the repository over time.

In its 2011 edition the course had 29 face-to-face sessions. Each session had a set of previous and in-class activities all designed using a template in Docbook including student material, instructor guide and solutions (if needed). The entire production flow is executed in less than five minutes in a regular lap-top computer. Changes in one file take a matter of seconds to propagate to the final production site.

As in the case of the first course, the amount of files, resources and authors makes the production management a task that could not be accomplished with the same level of quality without the support of Adagio. Each course activity contained a list of objectives, the required steps, a list of additional resources, and an embedded question about the time devoted by students. A set of auxiliary documents were created specifically for the topics included in the course. Half of the sessions were held in a lab and required various initial documents for students to extend. Adagio was used to create a specific rule to process each session.

Additionally, the content was adapted for two target groups: students and instructors. Using specific mark-up attributes embedded in the documents, authors could merge content within the

Deployment period	Jan. 09 – Dec. 11	HTML Documents	180
Final number of files	4257	Figures	142
Commits in 2009	1753	Style resources	90
Commits in 2010	1389	Auxiliary resources	36
Commits in 2011	1294	Total Resources	452
Total number of commits	4436		
Number of authors	10		

Table 2: Activity and resources for the Systems Architecture Course

4 A preview of the course material is available at http://www.it.uc3m.es/abel/as/info/syllabus_en.html

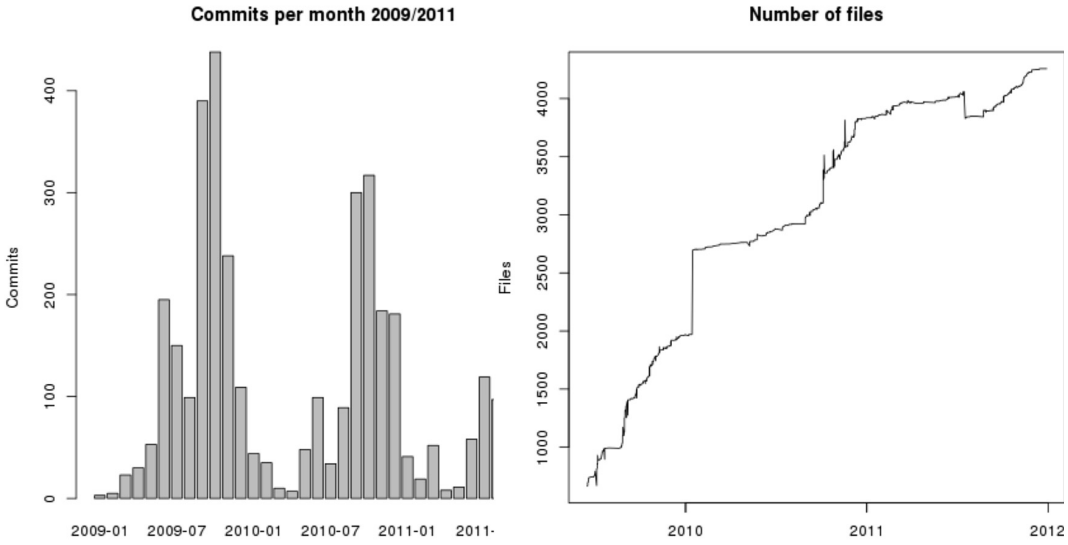

Figure 5: Three year evolution of the course material repository

same document for both versions. For example, after describing a step for an activity for the students, a description of potential pitfalls and aspects to take into account is included for the instructor following the single source approach described in Section 1. This technique can be easily extended for arbitrary groups by using the appropriate markup attributes. The previous and in-class activities were also included in a single document together with these instructor notes, and in two languages. Eight final documents (two for the previous activities, two with the in-class activities and their corresponding versions with the instructor guide) were automatically created and published.

The high number of activities, auxiliary files, text for the instructors, etc. made managing the production extremely complicated without the use of any automatic tool. The process was implemented using 70% of the rules offered by Adagio, mostly the processing of Docbook documents with XSLT. The hierarchy of rule files allowed for the common parts of the definitions to be shared among the entire site, and leave a simple personalization in the local rule files. The average length of these rule files is only 8.7 lines.

7.3 Concurrent Resource Management

A second measurement was conducted in the two scenarios to assess the level of concurrency achieved during the creation of the course resources using Adagio. Figure 6 shows the number of concurrent active authors creating or modifying course resources. A value of n means that there were n versions of the course repository containing changes made by users concurrently.

As it can be seen in the left side of the figure, the first course had periods where up to nine authors were manipulating the material in parallel. Analogously, the right side of the figure shows the activity for the second course. In this case, the number of concurrent versions peaks at six in the second and third course editions. Both scenarios clearly show how the adopted production paradigm allows for concurrent resource creation and management.

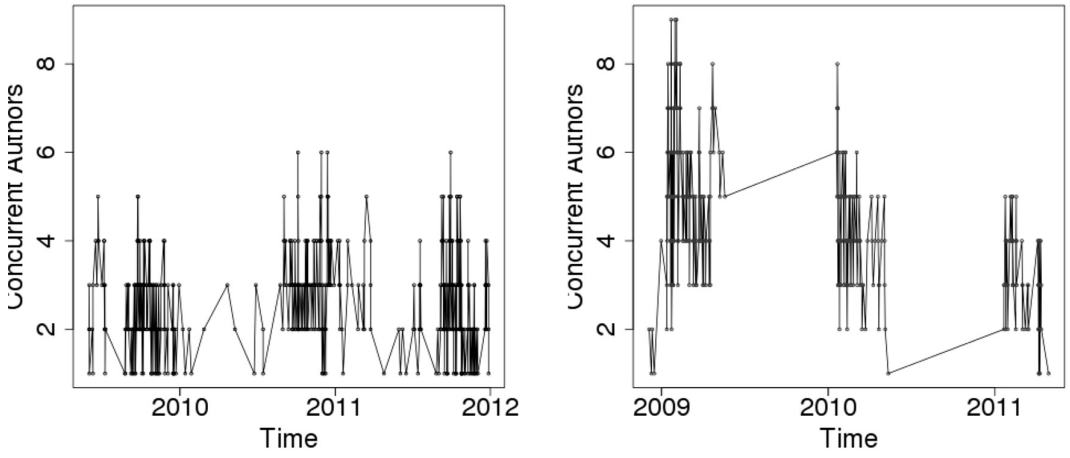

Figure 6: Number of concurrent authors over time

8. Conclusions and Future Work

The conditions under which learning content is being created are changing rapidly. The complexity derived from the highly interactive, multimedia-based content required, together with the adoption of collaborative authoring, the need for continuous updates, the re-use of material, and the adaptation to different target groups, lead to a new set of requirements for authoring environments. Existing solutions focus on supporting the author at the level of a single resource. New solutions are needed to embrace the production process at a higher level of abstraction to manage large numbers of resources by a community of authors.

In this document the Adagio authoring system has been described. In Adagio, a community of users collaboratively creates learning resources. These resources are then automatically processed, combined and published by executing a pre-defined set of production rules that process these documents and produce the resulting resources. The rules assume a single source approach where the required data for various resources are all included in a document from which multiple versions are obtained. Adagio offers a comprehensive set of XSL transformations that process documents in the Docbook format. Authors create a single document containing information to be shown to different stakeholders, in different contexts, with different elements, etc.

The set of pre-defined rules, together with some additional mechanisms, allow the system to support continuous update of learning resources, re-use typical production steps across large projects, and lower the barrier to adapt the material to different needs such as language, role, media format, etc. Also, the system has been gradually enhanced with various extensions to support emerging requirements derived from real-life scenarios. These extensions have been shown to easily accommodate new requirements as well as the integration of heterogeneous data sources derived from other auxiliary tools. The tendency to obtain and distribute content from an increasingly large set of platforms makes Adagio ideal to quickly adapt content to these scenarios.

The system has been validated by analyzing its use in two learning scenarios with a large number of authors over a three year period. Various indicators clearly show how the high complexity of the authoring process is significantly mitigated by using Adagio to deploy a robust processing

environment that is sustainable over time. The repositories of course material have been increasing steadily during this time period, showing no limitations. The production flow captured with Adagio is efficiently executed with the given rules. Additionally, the integration of Adagio with the version control system Git facilitated the concurrent creation of resources among a high number of authors, as well as the addition of new authors to the teams.

There are several future lines of work that are being explored. A more thorough process to capture user requirements for this type of platform might offer valuable hints as to where the system should be improved. Additionally, a preliminary study of the use of Adagio to create Learning Designs was conducted during the last edition of the first course. The experience showed how certain aspects of the process could be automated but additional functionality was needed to fully automate the translation from an arbitrary set of resources to a script and its description in a Unit of Learning.

Some Learning Management Systems (LMS) offer WebDAV support for managing resources, so a tighter integration with Adagio in which files are automatically uploaded to the course spaces is possible. This integration is desirable to reduce the turn-around time when updating content. Support for assessment and learning analytics are another two aspects that can be improved. Although the toolkit provides support for the creation and management of exams, some additional aspects of the process can be automated. When considering web-based assessment, resources can be created so that results can be automatically uploaded to the corresponding LMS. Also, the single source document approach is ideal to deploy functionality to capture student behaviour and apply learning analytics techniques to improve the overall learning process.

Acknowledgements

Work partially funded by the EEE project, "Plan Nacional de I+D+I TIN2011-28308-C03-01", and the "Emadrid: Investigación y desarrollo de tecnologías para el e-learning en la Comunidad de Madrid" project (S2009/TIC-1650).

References

BARAHONA, J., CHAPARRO, D., DIMITROVA, V., TEBB, C. and MAZZA, R. (2005): Producing educational resources in the "libre way": The Edukalibre project, in *International Conference on WWW*, Citeseer, 69–76.

BAUDRY, A., BUNGENSTOCK, M. and MERTSCHING, B. (2004): Reusing document formats for modular course development, in *Web-Based Education: Proceedings of the IASTED International Conference (WBE-2004)*, Acta Press Inc, # 80, 4500-16 Avenue N. W, Calgary, AB, T3B0M6, Canada, 535–537.

BROOKS, Jr., F.P. (1995): *The mythical man-month*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

BULLARD, L.G. and FELDER, R.M. (2007): A Student-centered approach to teaching material and energy balances 1. Course DEsign, *Chemical Engineering Education* 41(2): 93–100.

BUNGENSTOCK, M., BAUDRY, A. and MERTSCHING, B. (2002): The construction kit metaphor for a software engineering design of an e-learning system, in *Proc. ED-MEDIA*, 216–217.

CHACON, S. (2009): *Pro Git*, Apress, Berkeley, CA.

COOMBS, J.H., RENEAR, A.H. and DEROSE, S.J. (1987): Markup systems and the future, *Communications of the ACM* 30(11): 933–947.

DHOLAKIA, U., KING, W. and BARANIUK, R. (2006): What makes an open education program sustainable? The case of connexions, in *Open Education Conference*, Vol. 2340.

ERRINGTON, E. (2004): The impact of teacher beliefs on flexible learning innovation: some practices and possibilities for academic developers, *Innovations in Education and Teaching International* 41(1): 39–47.

FISLER, J., BLEISCH, S. and NIEDERHUBER, M. (2005): Development of sustainable e-learning content with the open source eLesson Markup Language eLML, in *ISPRS Workshop June 2nd/3rd 2005*.

FISLER, J. and SCHNEIDER, F. (2009): Creating, handling and implementing e-learning courses using the Open source tools OLAT and eLML at the University of Zurich, in *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 1.

FOWLER, M. (2006): (Accessed April 12, 2012), Continuous integration, http://www.martinfowler.com/articles/continuousIntegration.html.

GONZÁLEZ-BARAHONA, J.M., TEBB, C., DIMITROVA, V., CHAPARRO, D. and ROMERA, T. (2005): Transfering Libre software development practices to the production of educational resources : The Edukalibre project, in *International Conference on Open Source Systems*, number 110330.

GONZÁLEZ TÉLLEZ, A. (2010): E-learning authoring with Docbook and SMIL, in *Proceedings of the 2010 10th IEEE International Conference on Advanced Learning Technologies*, IEEE Computer Society, 246–248.

HADZILACOS, T., KALLES, D. and POULIOPOULOU, M. (2007): On the software and knowledge engineering aspects of the educational process, *International Journal of Software Engineering and Knowledge Engineering* 17(5): 643–662.

HENRY, G. (2004): Connexions: An alternative approach to publishing, in *European Conference on Digital Libraries. Research and Advanced Technology for Digital Libraries*, Springer, 421–431.

MARTÍNEZ-ORTIZ, I., MORENO-GER, P., SIERRA-RODRÍGUEZ, J.L. and FERNÁNDEZ-MANJÓN, B. (2006): Using DocBook and XML technologies to create adaptive learning content in technical domains, *International Journal of Computer Science & Applications* 3(2): 91–108.

MEISZNER, A., GLOTT, R. and SOWE, S.K. (2008): Free/Libre Open Source Software (FLOSS) communities as an example of successful open participatory learning ecosystems, *UPGRADE* 9(3): 62.

MOLLOY, D. (2003): Single-source interactive and printed content publishing using the Docbook XML standard, in *2nd International Conference on Multimedia and Information & Communication Technologies in Education*, Badajoz, Spain, Citeseer.

NOVAK, G.M., PATTERSON, E.T., GAVRIN, A.D. and CHRISTIAN, W. (1999): *Just in Time Teaching*, Prentice Hall, Upper Saddle Rive, NJ.

O'NEAL, A.F. (2008): The current status of instructional design theories in relation to today's authoring systems, *British Journal of Educational Technology* 39(2): 251–267.

PULICHINO, J. (2005): The content authoring research report 2005, Technical Report October, The eLearning Guild Research.

RAWLINGS, A., VAN ROSMALEN, P., KOPER, R., RODRÍGUEZ-ARTACHO, M. and LEFRERE, P. (2002): Survey of Educational Modelling Languages (EMLs), version 1, Technical report, CES/ISSS WS LT.

SIERRA, J.L., FERNÁNDEZ-VALMAYOR, A. and FERNÁNDEZ-MANJÓN, B. (2006): A document-oriented paradigm for the construction of content-intensive applications, *The Computer Journal* 49(5): 562–584.

SIERRA-RODRÍGUEZ, J.L., FERNÁNDEZ-MANJÓN, B., FERNÁNDEZ-VALMAYOR, A. and NAVARRO, A. (2005): Document-oriented development of content-intensive applications, *International Journal of Software Engineering and Knowledge Engineering* 15(6): 975–993.

STAYTON, B. (2007): *DocBook XSL: The Complete Guide*, 4th edn, Sagehill Enterprises, USA.

THOMAS, L. and RAS, E. (2005): Courseware development using a single-source approach, in *Proceedings of the World Conference on Education Multimedia, Hypermedia and Telecommunications, Ed-Media*, 4502–4509.

THOMAS, L. and TRAPP, S. (2007): Building re-configurable blended-learning arrangements, in *Balkan Conference in Informatics*, 271–280.

TRIDGELL, A. and MACKERRAS, P. (1996): The rsync algorithm, Technical report, Technical Report TR-CS-96-05, Australian National University.

VAN ROSSUM, G. (Last accessed: 2011): Python programming language. URL: http://www.python.org

WALSH, L. (2007): Using Extensible Markup Language (XML) for the single source delivery of educational resources by print and online : A case study, *Journal of the Association for the Advancement of Computing in Education* 15(4): 389–411.

Biographical Notes


Abelardo Pardo

Abelardo Pardo *recevied his MSc degree in computer science from the Polytechnic University of Catalonia, Spain, in 1991, and the PhD degree in computer science from the University of Colorado at Boulder in 1997. He is an associate professor of telematics engineering at the Carlos III University of Madrid. He has participated in numerous research projects both at international and national levels and is author of more than 100 publications in conferences and journals. He also has more than three years experience working for the computer industry in companies such as Mentor Graphics Inc. and AT&T Bell Labs. His current research interests are in technology enhanced learning, intelligent tutors, adaptation, computer supported collaborative learning, learning analytics and technology enhanced tutoring strategies.*


Jesus Arias Fisteus

Jesus Arias Fisteus *received a MSc degree in telecommunication engineering from the University of Vigo, Spain, in 2001, and a PhD degree in communication technologies from the Carlos III University of Madrid, Spain, in 2005. In his PhD thesis he proposed a formalism that uses model-checking for verifying business processes. His current research interests are the semantic web, linked data and content distribution in the internet. Currently he works as an assistant professor at the Carlos III University of Madrid.*


Carlos Delgado Kloos

Carlos Delgado Kloos *received the PhD degree in computer science from the Technical University of Munich and in telecommunications engineering from the Technical University of Madrid. He is full professor of telematics engineering at the University Carlos III of Madrid, where he is the director of the Nokia Chair and of the GAST research group. He is also vice-rector of Infrastructures and Environment. His main interests include internet-based applications and in particular e-learning. He has been involved in more than 20 projects with European (Esprit, IST, @LIS, eContent Plus), national (Spanish Ministry, Region of Madrid) and bilateral (Spanish-German, Spanish-French) funding. He has published more than 200 articles in national and international conferences and journals. He has further written a book and co-edited five. He is the Spanish representative at IFIP TC3 on Education and Senior Member of IEEE.*